LAMP-TR-057
CAR-TR-953
CS-TR-4186

MDA 9049-6C-1250
September 2000

# A New MERIT Version for MPEG-2 Encoded Files

Alain Pagani

Center for Automation Research
University of Maryland
College Park MD, 20742-3275

Ecole Centrale de Lyon
69130 Ecully, France

## Abstract

The *MPEG Encoded Retrieval and Indexation Toolkit* (MERIT) performs video segmentation of MPEG files in the compressed domain, using an algorithm based on macroblock type statistics. It was written in C by V. Kobla in 1995 for his doctoral work at the University of Maryland. Kobla's code dealt with MPEG-1 files only. In this report we update MERIT to analyze MPEG-2 files as well. We modify the file parsing process to account for the MPEG-2 specifications. We account for the new motion compensation modes introduced by MPEG-2 in preliminary computations, generating data structures that allow the original MERIT segmentation algorithm to work properly. A series of tests confirmed the validity of our solutions. The new version 4.0 of MERIT is a superset of MERIT 3.3, insofar as it gives the same results for MPEG-1 files, and is able to analyze MPEG-2 files using almost all the available options. Further improvements could be made to address key-frame storage and higher chrominance formats.

**Keywords:** video segmentation, MERIT, MPEG specifications, macroblock, motion compensation.

This research was funded in part by the Department of Defense and the Army Research Laboratory under Contract MDA 9049-6C-1250. Thanks to Sara Larson for formatting this report.

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED |
|---|---|---|
| **SEP 2000** | | **00-09-2000 to 00-09-2000** |

| 4. TITLE AND SUBTITLE | 5a. CONTRACT NUMBER |
|---|---|
| **A New MERIT Version for MPEG-2 Encoded Files** | 5b. GRANT NUMBER |
| | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | 5d. PROJECT NUMBER |
| | 5e. TASK NUMBER |
| | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275** | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES
**The original document contains color images.**

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | |
| **unclassified** | **unclassified** | **unclassified** | | **42** | |

LAMP-TR-057
CAR-TR-953
CS-TR-4186

MDA 9049-6C-1250
September 2000

# A New MERIT Version for MPEG-2 Encoded Files

Alain Pagani

# A New MERIT Version for MPEG-2 Encoded Files

Alain Pagani

Center for Automation Research
University of Maryland
College Park MD, 20742-3275

Ecole Centrale de Lyon
69130 Ecully, France

**Abstract**

The *MPEG Encoded Retrieval and Indexation Toolkit* (MERIT) performs video segmentation of MPEG files in the compressed domain, using an algorithm based on macroblock type statistics. It was written in C by V. Kobla in 1995 for his doctoral work at the University of Maryland. Kobla's code dealt with MPEG-1 files only. In this report we update MERIT to analyze MPEG-2 files as well. We modify the file parsing process to account for the MPEG-2 specifications. We account for the new motion compensation modes introduced by MPEG-2 in preliminary computations, generating data structures that allow the original MERIT segmentation algorithm to work properly. A series of tests confirmed the validity of our solutions. The new version 4.0 of MERIT is a superset of MERIT 3.3, insofar as it gives the same results for MPEG-1 files, and is able to analyze MPEG-2 files using almost all the available options. Further improvements could be made to address key-frame storage and higher chrominance formats.

**Keywords:** video segmentation, MERIT, MPEG specifications, macroblock, motion compensation.

# 1 Overview

MERIT (*MPEG Encoded Retrieval and Indexing Toolkit* [2, 11]) is a package that performs various MPEG-based analyses, such as video segmentation [5, 10], motion analysis, extraction of estimated DC coefficients or of flow vector information. It was written in C by V. Kobla for his doctoral work at the University of Maryland. Video segmentation consists of identifying breaks or cuts in an MPEG-encoded video clip, thus dividing the video into shots. A shot can be defined as a minimal sequence of frames resulting from a continuous uninterrupted recording by an input device such a camera. In MERIT, the analysis is performed in the compressed domain using available macroblock and motion vector information, and, if necessary, DCT information. When MERIT is run without any options and with just the MPEG filename, it prints to *stdout* the list of key-frame numbers, which are the first frames of the segmented shots. These key-frames can be used in further applications, such as archiving or indexing of video sequences [3, 4]. Options available with MERIT will be described in Section 2.2.

MPEG is a digital compression standard for both video and audio, developed by the ISO MPEG committee. ISO is developing a family of MPEG standards. The group produced MPEG-1, the standard for Video CD and MP3; MPEG-2, the standard for Digital Television set top boxes and DVD; and MPEG-4, the standard for multimedia on the Web. It is also developing the MPEG-7 "Multimedia Content Description Interface".

Until now, MERIT could extract information only from MPEG-1 encoded files. An error would occur if one tried to read an MPEG-2 encoded file with MERIT. The main goal of this project consisted in developing a new version of MERIT which is able to process MPEG-2 encoded files. Actually, MPEG-2 is a superset of MPEG-1. This means that the MPEG-2 standard includes the MPEG-1 standard. Thus, an MPEG-2 version of MERIT works with MPEG-1 files as well.

In this report, the current MERIT 3.3 program is first outlined in detail. A description of the specific MPEG syntax used in this report can be found in Appendix A. Then, we describe what needs to be removed or changed in Version 3.3 of MERIT. Finally, we report on the implementation of the new version, the problems encountered, and their solutions.

# 2 MERIT 3.3

## 2.1 System overview

Compressed video in MPEG format is first segmented into shots (continuous uninterrupted sequences of frames). The segmented video is then passed on to a motion vector analysis routine where global camera motion information is extracted. Using the camera motion information, shots are subdivided into subshots. Finally, key-frames are extracted from these shots and subshots.

A shot change usually occurs abruptly between two frames. Such a shot change is called a cut. Cut detection is performed using the macroblock (MB) type. If a P-frame contains primarily intra-coded MBs, this suggests that these MBs could not be predicted from the previous reference frame, and a cut may have occurred between the

previous I- or P-frame and the current P-frame. If in a B-frame a majority of MBs are forward-predicted from the previous I- or P-frame, there is a high probability that a shot change has occurred between the current frame and the next I- or P-frame. Similarly, if a majority of MBs in a B-frame are backward-predicted, there is a high probability that a shot change has occurred between the previous I- or P-frame and the current frame. If there is a shot change, all B-frames before the cut must have a majority of forward-predicted MBs, and all B-frames after the cut must have a majority of backward-predicted MBs. Tests for the presence of these features are employed in the detection process.

When the macroblock information is found to be inconclusive, the algorithm uses the DCT coefficients to confirm the existence of shot changes. After the shots have been found, MERIT uses the motion vector information to find subshots, or scene changes. A paper written by V. Kobla [5] provides details about this algorithm.

### 2.2 Usage

Many options are available with MERIT [2]. When MERIT is run without any options and with just the MPEG filename, it prints to stdout the list of key-frame numbers, which are the first frames of the segments defined by the breaks. Depending on the option, the algorithm uses the I-frames for segmentation analysis (`-dct`), or performs DCT validation for certain (`-valid`) or all (`-full`) detected cuts. The motion analysis can also be turned on (`-motion`). A display window can provide graphic information about the MB types and motion vectors (`-dispMV`, `-dispFlow`). Moreover, several types of information can be printed or saved in a file. Finally, one can choose the start-frame and the end-frame numbers (`-start`, `-end`).

### 2.3 General architecture

MERIT comprises two main components: bit stream parsing and video segmentation. The first component parses the MPEG file and collects the relevant information in specific structures. The second component uses this information to perform video segmentation. As we will see, these two parts are quite independent.

#### 2.3.1 MPEG bit stream parsing

The `main()` function is located in the `MERIT.c` file. This function processes the options and calls `Parseblkfile()` or `ParseDctFile()`, both located in the `Parse.c` file. These two functions are intermediate functions, insofar as they only set options for another function that they call, `mpeg_stat()`, located in `main.c`.

In fact, `mpeg_stat()` is the function which parses the stream. It takes two arguments. The first, `mpegfilename`, is the name of the file to be parsed. The second, `dct`, is a flag indicating whether the DCT coefficients are needed (1) or not (0). The function `mpeg_stat()` corresponds to the `main()` function of another program, called *mpeg_stat* and written by Steve Smoot at U. C. Berkeley. The MERIT program includes all files from mpeg_stat, some of which have been adapted by V. Kobla (`filter.c`, `main.c`, `parseblock.c`, `proto.h`, `util.c`, `video.c`, `video.h`). These imported files can easily be

distinguished from the MERIT-specific files, since they begin with a lower-case letter (e.g. `video.c`), whereas the MERIT-specific files begin with a capital letter (e.g. `Parse.c`).

The main purpose of the original mpeg_stat is to provide statistics about the features of an MPEG-1 encoded file. Therefore it is mainly a parsing program, and it is used by MERIT. A detailed description of mpeg_stat is provided below. The parsing function of MERIT is completely performed by the mpeg_stat code, and only uses some global variables shared by other parts (defined in `Global.h` and `List.h`). Actually, mpeg_stat has been modified to fill in specific global structures with information, instead of passing it through files, as in the original version.

### 2.3.2 Storing MPEG information

In order to run the segmentation algorithm, MERIT needs some preliminary information from the MPEG file. Here is a short description of the structures in which the function `mpeg_stat()` stores the information. These structures are declared and defined in `Global.h`, `List.h` and `List.c`

- `List.h`

  This file declares the `Line` structure as follows:

```
typedef struct LINE
{
      int LineType,blockType,frameNo,qscale,numBits,blockNo;
      char frameType,*dctSpecs;
      MV *forw,*back;
      struct LINE *next;
} Line;
```

  `LineType` is the type of the Line, which can be `BLOCK` (value: 10), `SLICE` (11), `FRAME` (12) or `GOP` (for Group Of Pictures, value: 13).

  `blockType` is the type of the block, if the `LineType` is `BLOCK`. Possible values: `INTRA` (200), `FORW` (201), `BI` (202), `BACK` (203), or `SKIP` (204).

  `frameNo` is the number of the current frame.

  `qscale` is the quantization scale used for the DCT.

  `numBits` is the total number of bits used for a particular MB.

  `blockNo` is the number of the macroblock, when needed (if `LineType` = `BLOCK` only).

  `frameType` is the type of the frame, when needed (`'I'`,`'P'`, or `'B'`, if `LineType` = `FRAME` only).

  `dctSpecs` is a string with all the DC and AC coefficients for each block of a macroblock (if `LineType` = `BLOCK` only).

  `forw` is the forward motion vector (`MV` is a structure with the $x$ and $y$ coordinates), when needed (if `LineType` = `BLOCK` only).

`back` is the backward motion vector, when needed (if `LineType` = `BLOCK` only).

Initially, the `Line` information was to be an element of a linked list, as indicated by the presence of a pointer to the next `Line`. The function `add_node()` adds a new empty current node to the list. But actually the program does not use this list structure. The function `add_node()` is never called and the information is stored in a current `Line` and immediately processed, before the current `Line` is refreshed (`refresh_node()`) and new information is stored.

- `List.c`

  This file defines `head` and `current` as pointers to `Line`, and defines the function `add_node()` (never called), and the function `refresh_node()`. This function resets the `Line` node pointed to by `current` by setting all the integers or char to 0 and the pointers to `NULL`. If the `current` pointer is null, it creates an empty structure for `Line` and stores its address in `current`. The `head` pointer is never used, since only the `refresh_node()` function is called. `current` points to the `Line` structure where all the needed information found in the MPEG bit stream is stored.

- `Global.h`

  `Global.h` defines some helpful integer constants for the understanding of the code (such as `BLOCK = 10`, `SKIP = 204`...). It also declares the function `mpeg_stat()`, and some useful structures, such as `MV` (motion vector, with the $x$ and $y$ coordinates in half-pixels; brightness values between pixels are computed using bilinear interpolation), `flMV` (float motion vector, for averaging). Furthermore, `Global.h` defines some structures where the information is stored just after parsing and temporary storage in the `current Line` structure. The most important structures are `Info` and `InfoList`

```
typedef struct INFO
{
      int dispFrameNo;
      char frameType;
      int value;
      int numIntra,numForw,numBack,numBidir,numSkip;
      int numSkipForw,numSkipBack,numSkipBidir;
      MV *forwMVs;
      MV *backMVs;
      MV *domMV;
      flMV *meanMV;
      int avgAngle;
      char *MBTypes;
} Info;


typedef struct INFOLIST
{
```

```
    Info *info;
    int frameNo;
    struct INFOLIST *next;
} InfoList;
```

As we see, all the information about a frame is stored in the `Info` structure (frame number, frame type, number of Intra-coded MBs, number of Forward-, Backward-, and Bidirectionally-predicted MBs, number of all types of Skipped MBs, ...).

This `Info` for each frame is then put into a list structure (`InfoList`).

The main difference between the `Line` structure and the `Info` structure is that `Line` can be used to describe different levels of objects in a video stream, such as Group of Pictures, Pictures, Slices, MBs and Blocks, depending on the `LineType`. On the other hand, `Info` contains information for a whole frame (picture), such as the frame type and the number of macroblocks of each type. This structure is used in the segmentation algorithm, which needs to compare the features of successive frames.

Other structures defined in `Global.h` (`MotionInfo`, `SceneInfo`, `RangeList`, ...) are used in the segmentation algorithm. However, we don't need to study these structures, since our upgrade of MERIT only required a modification of the parsing component

To insert the information stored in the `current Line` into `Info`, and then into the `InfoList` structure, MERIT uses functions mainly referred to as *initInfo()*, and defined in `Parse.c`. *initInfo()* has one argument, a pointer to a `Line` structure. Essentially, *initInfo()* deals with `current`. After each parsing by mpeg_stat (that is, when `current` is full), a call to *initInfo(*`current`*)* puts the information into the `InfoList` structure. Then, `current` is refreshed.

There are three *initInfo()* functions: `initInfoBlkMode()`, `initInfoDctMode()`, and `initInfoDCTValidMode()`. Only one of them is used each time MERIT is running, depending on the current mode (options of the program).

### 2.3.3 Video segmentation

When the parsing is completed, the video segmentation part of MERIT begins its work. In fact, the segmentation is totally independent of the parsing, once the MPEG bit stream has been read. Here is a description of all the information MERIT 3.3 currently needs for the video segmentation:

- Video Stream level:
  - horizontal picture size in pixels (`h_size`)
  - vertical picture size in pixels (`v_size`)
  - horizontal picture size in MBs (`mb_width`)
  - vertical picture size in MBs (`mb_height`)
  - total number of frames in the stream (`totalFrames`)
  - total number of I-frames (`numIFrames`)

- Frame level:
  - frame number (display order)
  - frame type (`'I'`, `'P'`, or `'B'`)
- Macroblock level:
  - macroblock number in the frame (0 to `mb_width` $\times$ `mb_height` $-$ 1)
  - macroblock type (`FORW`, `BACK`, `INTRA`, `BI`, `0` (forward and no motion vector), `SKIP`)
  - macroblock motion vectors(s)
  - macroblock quantization scale
  - number of bits used for this MB (`numBits`)
- Block level (only if the DCT validation is needed):
  - DCT quantized DC coefficient (0 to 2040, for intra-coded MBs only)
  - DCT quantized AC coefficients : position in matrix (0 to 63) and level ($-2047$ to 2047) of non-zero coefficients

Four modes of segmentation are provided by MERIT: blockinfo mode (MERIT's simplest use, without any option), dct mode (`-dct`), validation mode (`-valid`), and full mode (`-full`).

When running in blockinfo mode, MERIT uses only the MB_type information. In fact, for each frame a variable called `value` is computed. Its numerical value depends on the picture coding type (I, P, or B), but is set as the number of MBs in the frame that are of a certain MB type. Therefore `value` is in the range between 0 and the maximum number of MBs in the frame. Usually, `value` is high for pictures very similar to the previous reference frame(s), and low for pictures with no similarities to the previous reference frames(s). An algorithm then uses this `value` and finds the cuts.

With the dct mode, MERIT only uses the I-frames. This mode is useful for streams in XING format (only I-pictures). For each block of a specific I-frame, the DC coefficient (first and dominating coefficient of the cosine transform of the block) is compared to the DC coefficient of the same block in the previous I-frame. If the difference is higher than a fixed threshold, a variable called `sdd` is increased. Finally, `sdd` is the number of non-similar blocks of two consecutive I-frames, and reflects the difference between these I-frames, and therefore the likelihood of a cut between these frames.

In the validation mode, verification is used after a first pass in blockinfo mode to validate the cuts, when there are ambiguities. The decision whether to perform validation depends on a fixed threshold of skipped macroblocks in the frame. If the number of skipped macroblocks is higher than this threshold, the number of significant macroblocks is found to be insufficient and the validation is performed. For each ambiguous cut, an algorithm finds the first previous I-frame and the first next I-frame. Then the `dct` mode

is performed for these two frames only. MERIT verifies the presence of a cut and decides whether to keep or reject this cut.

The full mode is similar to the validation mode, except that the dct validation is performed for all the cuts, even if there is no ambiguity.

### 2.3.4 Other tools

MERIT also provides other tools, described by V. Kobla as *secondary options for performing specific minor tasks* [2].

**The `-dispMV` and `-dispFlow` options** pop up a window displaying various information about the macroblocks of each frame. dispMV shows the macroblock types and the motion vectors, and dispFlow shows the flow vectors. The flow vectors can be computed and stored without display, using the option `-flow`.

**Motion analysis** can be turned on with the `-motion` option. This subshot analysis is based on the extracted camera motion information.

**Estimation of the DC coefficients** of P and B frames is possible with `-estimateDC`, in different color spaces.

**The `-saveppms` and `-showKey` options** allow the user to store and display the key-frames. While `-saveppms` only stores the key-frames in a *ppm* subdirectory, `-showKey` creates and displays a montage of the key-frames, using *ImageMagick* utilities with the stored key-frames.

**Content subshot analysis** can be performed with `-contentAll` and `-contentMotion`. `-contentAll` uses the flow of all frames whereas `-contentMotion` uses the flow of frames in consistent camera motion sequences only.

## 3 mpeg_stat and mpeg2stat

According to what we have seen in earlier sections, in order to make MERIT compatible with MPEG-2, we mainly need to upgrade the parsing component of MERIT. That is, we need to understand how mpeg_stat operates, and how and when the `current Line` structure is updated. A new version of mpeg_stat, mpeg2stat, already exists and is able to analyze MPEG-2 files. We have to modify mpeg2stat in the same way as mpeg_stat has been modified for MERIT. Thus, an `mpeg2stat()` function in MERIT must store information in the same structures as `mpeg_stat()` did. It is not necessary to touch the video segmentation part. Therefore, we now focus on describing the parsing program mpeg_stat, its adaptation to MERIT, and the latest version of mpeg_stat, mpeg2stat.

### 3.1 mpeg_stat

mpeg_stat is used to provide statistics from an MPEG-1 file. Therefore, it parses the MPEG bit stream, and collects information. In its original version, the output can be stored in specific files. Some options are available, listed below.

```
-quiet:                 Turn off output of frame types/matrices as
                        encountered
-verify:                Do more work to help assure the validity of
                        the stream
-start N:               Begin collection at frame N (first frame is 1)
-end N:                 End collection at frame N (end >= start)
-histogram file:        Put detailed histograms into file
-qscale file:           Put qscale information into file
-size file:             Write individual frame type and size into file
-offsets file:          Write high-level header offsets into file
-block_info file:       Put macroblock usage into file
-dct                    Put decoded DCT information into block file
-rate file:             Put instantaneous rate information in file
-ratelength N:          Measure bit rate per N frames, not one
                        second's worth
-syslog file:           Store parsing of system layer into file
-userdata file:         Store user data information into file
-time:                  Measure time to decode frames
-all file:              Put all information into files with basename
                        file
```

When mpeg_stat is run without any options and just with an MPEG filename, it simply parses the stream and outputs information to stdout.

The `main()` function is located in the `main.c` file. This function only processes the options and calls another function for the parsing: `mpegVidRsrc()`, located in the `video.c` file. This file seems to be the main file for the parsing. It contains several useful functions such as `ParseGOP()` and `ParsePicture()`.

### 3.1.1 Storing structures

While reading the bit stream, mpeg_stat puts the information into a specific structure similar to the `Line` structure in MERIT. This structure is called `VidStream`, and is defined in `video.h`. Here is a description of this video stream structure, with some explanations. The `GoP` (Group of Picture), `Pict` (Picture), `Slice`, `Macroblock` and `Block` structures are defined in `video.h` as well.

```
/* Video stream structure.  */

typedef struct vid_stream
{
unsigned int h_size;                    /* Horiz. size in pixels. */
unsigned int v_size;                    /* Vert. size in pixels. */
unsigned int mb_height;                 /* Vert. size in mblocks. */
unsigned int mb_width;                  /* Horiz. size in mblocks. */
unsigned char aspect_ratio;             /* Code for aspect ratio. */
```

```
unsigned char orig_picture_rate;          /* Code for picture rate. */
unsigned char picture_rate;               /* A valid picture rate. */
unsigned int bit_rate;                    /* Bit rate. */
unsigned int vbv_buffer_size;             /* Minimum buffer size. */
BOOLEAN const_param_flag;                 /* Constrained parameter
                                          flag. */
unsigned char intra_quant_matrix[8][8];   /* Quant. matrix for intracoded
                                          frames. */
unsigned char non_intra_quant_matrix[8][8];/* Quant. matrix for non-
                                          intracoded frames. */
char *ext_data;                           /* Extension data. */
char *user_data;                          /* User data. */
int ext_size;                             /* Length of Extension data */
int user_size;                            /* Length of User data */
GoP group;                                /* Current group of pictures */
Pict picture;                             /* Current picture. */
Slice slice;                              /* Current slice. */
Macroblock mblock;                        /* Current macroblock. */
Block block;                              /* Current block. */
int state;                                /* State of decoding. */
int bit_offset;                           /* Bit offset in stream. */
unsigned int *buffer;                     /* Pointer to next byte in
                                          buffer. */
int buf_length;                           /* Length of remaining
                                          buffer. */
unsigned int *buf_start;                  /* Pointer to buffer start. */
int max_buf_length;                       /* Max length of buffer. */
PictImage *past;                          /* Past predictive frame. */
PictImage *future;                        /* Future predictive frame. */
PictImage *current;                       /* Current frame. */
PictImage *ring[RING_BUF_SIZE];           /* Ring buffer of frames. */
} VidStream;
```

As we can see, like the List structure in MERIT, this VidStream structure follows the MPEG specifications about the semantics of the video stream encoding. The video stream is divided into Groups of pictures, Pictures, Slices, Macroblocks and Blocks.

### 3.1.2 Parsing techniques

The `mpegVidRsrc()` function looks at the first bits of the current bit stream. Then it identifies one of the following codes: *Sequence-start-code*, *Sequence-end-code*, *GOP-start-code*, *Picture-start-code*, *Slice-start-code*, *Macroblock-start-code* or *Block-start-code*. Then it calls a specific function to parse the stream, according to the code found: `ParseSeqHead()`, `ParseGOP()`, `ParsePicture()`, etc. This process is repeated until the end of the bit stream is reached (or the last specified frame, if the `-end` option is used).

All the *Parse"type"()* functions are defined in `video.c` except for the block type parsing function, which is defined in the `parseblk.c` file.

Typically, a *Parse"type"()* function parses the features of the *type* and fills in the `VidStream` structure. At the same time, it writes some information into files if needed.

### 3.2 Adaptation of mpeg_stat for MERIT

When the mpeg_stat program was incorporated into MERIT, some modifications were needed. We now discuss the relevant modifications that were required by MERIT.

#### 3.2.1 mpeg_stat()

The `main()` function of mpeg_stat has been renamed `mpeg_stat()` in MERIT. While `main()` does the processing of the options, `mpeg_stat()` in MERIT only requires the mpeg filename and a `dct` flag. Thus, the modifications are only an adaptation of the option processing. Then, like the `main()` function in mpeg_stat, `mpeg_stat()` calls `mpegVidRsrc()`. The MERIT file `main.c` includes the new files

```
#include ``Global.h''
#include``List.h''
```

where it defines the extern `Line` pointers `head` and `current`. At the beginning and end of `mpeg_stat()`, these pointers are freed and filled with the `NULL` constant.

`mpegVidRsrc()` is located in `video.c`. This file seems to be the one with the most important modifications. First, there are declarations of external variables and some necessary internal variables or functions. Then, `mpegVidRsrc()` and its *Parse"type"()* subfunctions have been modified to fill the `current Line` while parsing. This filling will now be referred to as the *InfoList building algorithm*. It involves putting the bit stream information in the `current Line` and transferring the information from `current` to `Info` and `InfoList`.

#### 3.2.2 The InfoList building algorithm

Here is a description of the *InfoList building algorithm* which will be useful for the upgrade to MPEG-2 processing. The square brackets contain the name of the function where the action is performed, and its location.

```
/* extern variables */
extern int totalFrames
extern int numIFrames
extern int maxMV
extern int mpeg_mode
extern int pass1TotalFrames
extern Info **info
extern Line *current
int h_size,v_size,mb_width,mb_height;
```

```
/* extern functions */
extern void add_node() /* Actually no use is made of this function */
extern void refresh_node()
extern int parseDctSelectiveContd(Info **info, int totalframes)
extern void initInfoBlkMode(Line *current)
extern void initInfoDctMode(Line *current)
extern void initInfoDCTValidMode(Line *current)

/* useful intern variables */
MV *forw, *back
int mbtype

[mpegVidRsrc(), video.c]
while not(end of file)
read next_start_code
start:

        switch start_code

                case sequence_end_code:
                        exit

                case sequence_start_code:
                        [ParseSeqHead(), video.c]
                        parse sequence header
                        get horizontal size of image space (h_size)
                        get vertical size of image space (vs_size)
                        calculate macroblock horizontal size of image (mb_width)
                        calculate macroblock vertical size of image (mb_height)

                read next start code
                goto start

        case group_of_picture_code:
                [ParseGOP(), video.c]
                parse GOP header
                refresh_node()
                current->LineType=GOP
                initInfo()

                read next start code
                goto start

        case picture_start_code:
```

```
        [ParsePicture(), video.c]
        parse picture header
        refresh_node()
        current->LineType=FRAME
        current->frameNo= ‘current frame #’
        current->frameType= ‘current frame type’ (OIPB)
        totalFrames++
        if frameType=I numIFrames++
        initInfo()

        read next start code
        goto start

case slice_start_code
        [ParseSlice(), video.c]
        parse slice header
        refresh_node()
        current->LineType=SLICE
        initInfo()

        read next start code
        goto start

default:
        [ParseMacroBlock(), video.c]

        if (Previous Macroblocks skipped)
                {
                [ProcessSkippedPFrameMBlocks(), video.c]
                [and ProcessSkippedBFrameMBlocks(), video.c]
                for each skipped MB, do
                        {
                        refresh_node()
                        current->LineType=BLOCK
                        current->qscale=‘block quantization scale’
                        current->numBits=0
                        current->blockNo=‘current block #’
                        current->blockType=SKIP
                        initInfo()
                        }
                }

        mbtype=‘current MB type’ (FORW,BACK,INTRA,BI, 0)
        switch mbtype
                case FORW :
```

```
                forw->x='current right forw. mv'
                forw->y='current down forw. mv'
                update maxMV (all directions, if necessary)
                case BACK :
                back->x='current right backw. mv'
                back->y='current down backw. mv'
                update maxMV (all directions, if necessary)
                case BI :
                forw->x='current right forw. mv'
                forw->y='current down forw. mv'
                back->x='current right backw. mv'
                back->y='current down backw. mv'
                update maxMV (all directions, if necessary)
                default :

        refresh_node()
        current->LineType=BLOCK
        current->blockNo='current block #'
        current->blockType=mbtype
        current->qscale='current quantization scale'
        current->numBits='number of bits for the current MB'
        current->forw=forw
        current->back=back
        if (DCT info needed)
                 {
                 current->dctSpecs = 'current DCT Collection String'
                 }
        initInfo()

        read next start code
        goto start
```

Since the structure of mpeg2stat is nearly the same as the structure of mpeg_stat, this algorithm will be used in the adaptation of mpeg2stat to MERIT. Next, we have to understand how mpeg2stat works.

### 3.3 mpeg2stat

mpeg2stat is a software code written by Ian Gordon (©1998), with original intent to collect and output statistics from MPEG-1 and MPEG-2 files. It has been adapted from the MPEG Software and Simulation Group's decoder *mpeg2decode* (©MSSG 1996). It can use the information from a base layer file and from an enhancement layer file (scalability).

The statistics given by mpeg2stat are basically the same as in mpeg_stat. Moreover, depending on the verbose level (`-v` option), mpeg2stat displays MPEG-2 specific infor-

mation at different levels (Sequence, GOP, Picture, Slice, Macroblock). With the trace option (`-t`) the AC coefficients of each block are given.

There are no storing structures in mpeg2stat, since the decoded information is output on the fly, while parsing the bit stream. Again, mpeg2stat follows the MPEG specification structure, giving the information at different specified levels.

### 3.3.1 Parsing techniques

The `main()` function is located in `mpeg2dec.c`, and only processes the options. Then, in `Decode_Bitstream()` [`mpeg2dec.c`], a first call to `Header()` [`mpeg2dec.c`] finds out whether there is a video stream. `Header()` just calls another function, `Get_hdr()`, located in `gethdr.c`. This function recognizes the header type (sequence_start, GOP, picture, slice, sequence_end).

After the first call to `Header()`, `video_sequence()`[`mpeg2dec.c`] is the function where the routine begins. `Header()` is called until there is a picture header. If so, `Decode_picture()` [`getpic.c`] is called, then `Header()` again. This routine ends when `Header()` returns an *end-of-sequence* message. `Decode_picture()` basically calls several functions which collect all the information about each macroblock in the picture, including macroblock type, motion vectors [`getvlc.c`], and DCT coefficients of each block [`getblk.c`]. The architecture of this program is quite similar to that of mpeg_stat.

### 3.3.2 Modifications of mpeg2stat

To prepare mpeg2stat for MERIT, it was necessary to debug it and to add new options. The debugging consisted mainly of adding some outputs forgotten by the author. In particular, there was originally no output of DCT coefficients for MPEG-1 files, and no output of the DC coefficients used by MERIT. Furthermore, the way DCT coefficients are collected was changed. In the original program, the DCT coefficients were output on the fly while parsing the stream. In the new version, the DCT coefficients are collected in a string and output at the end of each macroblock parsing. This string, the `dctSpecifics` variable, is useful for the upgrade of MERIT, because it was produced by mpeg_stat and was used by the segmentation algorithm. After these changes mpeg2stat stores the DCT information the same way mpeg_stat does.

Finally, `-start` and `-end` options were added, which allow the user to collect statistics only for a specified range of frames. These two options are necessary for the validation option in MERIT.

### 3.3.3 Integration

Having created an upgraded version of mpeg2stat, we now have to integrate it into MERIT, just as mpeg_stat was. That is, the `main()` function needs to be modified, and the same *InfoList building algorithm* must be included in mpeg2stat.

# 4 MERIT 4.0

The first version of MERIT 4.0 was an alpha version whose purpose was to verify the possibility of integrating with mpeg2stat. We first focus on this version. Then some relevant MPEG-2 specifications are discussed, and the final version is outlined in more detail.

## 4.1 MERIT 4.0 alpha

The main idea of the alpha version is that a correct usage of the *InfoList building algorithm* would allow MERIT to perform its segmentation algorithm. This version indirectly uses mpeg2stat, insofar as it needs a parsing from mpeg2stat to produce an `info.t` file. This file is basically a copy of the standard output of mpeg2stat with the `-t` trace option obtained by redirection.

Example: `mpeg2stat -t filename.mpg > info.t`

`info.t` contains all the picture, macroblock and block information needed to perform block-based segmentation. For MERIT 4.0 alpha, all the functions dedicated to segmentation were kept and removed from mpeg_stat. A `main.c` file containing a new `mpeg_stat()` function was added to parse the `info.t` file and collect needed information. While parsing, information is sent to the segmentation part of MERIT, in accord with the *InfoList building algorithm.*

Example: `MERIT info.t`

This version sends only the macroblock type information, omitting the DCT segmentation. The results of the macroblock type algorithm are satisfactory, when tested with MPEG-1 files or MPEG-2 files coded with frame pictures. This first series of tests made it apparent that MPEG-2 specific picture structures such as *field pictures*, and MPEG-2 specific motion compensation modes such as *field prediction* or *Dual Prime mode*, would require preliminary computations before the original MERIT segmentation algorithm could be used. Another problem was the relatively small number of MPEG-2 video files available on the Internet, especially when looking for uncommon picture structures or motion compensation modes.

## 4.2 MPEG-2 specificity issues

As explained in Appendix A, MPEG-2 is a superset of MPEG-1. To be more precise, an MPEG-2 decoder is expected to be able to read MPEG-1 files. But an MPEG-2 encoder cannot write an MPEG-1 file, because of the MPEG-2 specifications. At best an MPEG-2 encoder can write an MPEG-2 file with the same characteristics as an MPEG-1 file (*frame pictures*, *frame prediction*, *frame dct* and *progressive sequence*). However, since MERIT has been written for MPEG-1 files, it cannot deal with MPEG-2 specifications. We chose to keep the segmentation part of MERIT as it was, for several reasons. First, this algorithm was written by others and it was risky to make changes to it. Second, an adaptation of the segmentation algorithm of MERIT could change the results for MPEG-1 files, which is not expected in an upgrade. This is the reason why mpeg2stat was used to parse the MPEG-2 video stream, and send information to MERIT in the same form that it would expect from an MPEG-1 file. With this solution, MPEG-1 files

are handled as in the previous version, and the integrity of MERIT's algorithm is not endangered.

Having chosen this solution, it became necessary to find how to handle MPEG-2 specifications. Here are the solutions that were chosen.

- For *frame pictures* with *frame prediction* and *frame dct*, the results are the same for MPEG-1 and MPEG-2 files (MPEG-1 uses only *frame pictures*, *frame motion compensation* and *frame dct*). For frame pictures with field prediction, MERIT uses an average of the two motion vectors corresponding to the two fields.

- *Field dct* gives the same results as *frame dct*, since MERIT uses only the DC coefficient of each block, which does not depend on `dct_type`.

- For *field pictures*, only the first field of a frame is used. This solution was adopted to avoid 'inter-field' prediction problems. According to the MPEG-2 specifications, a macroblock from the second field of a frame may be predicted using motion vectors with respect to the first field of the same frame. Macroblocks from the first field can be predicted only with respect to a field of another frame (i.e., not the second field of the same frame). Since the segmentation algorithm of MERIT is based on the macroblock inter-frame prediction type, the second field cannot be used without time-wasting computations. Each macroblock from the first field is counted twice to respect the total number of macroblocks. The two identical macroblocks are naturally placed one above the other. Since the macroblocks are coded in horizontal order first, the identical macroblock is stored in a buffer until the end of the macroblock line is reached. Then the stored macroblocks are processed and the buffer is emptied before the next macroblock line in the field is read. Again, in the case of $16 \times 8$ *MC prediction*, an average of the two motion vectors is computed and used as the unique motion vector of the macroblock.

- When *Dual Prime* prediction is used (see Section A.5.5), the motion vector is the *coded vector*. There is no use including the *dm vector*, since its length wouldn't significantly change the motion vector.

All these approximations of MPEG-2 files imply a loss of precision. But at the same time, the information seems to be sufficient according to the results of several series of tests, and using less information increases processing speed. The next section describes how these adaptations are implemented in MERIT 4.0.

### 4.3 MERIT 4.0 implementation

In this final version, all files from the segmentation part of MERIT (filenames beginning with a capital letter) were kept, and the mpeg2stat files (filenames beginning with a lower-case letter) were added. The mpeg2stat files were taken from the modified version of mpeg2stat. None of the files of the segmentation component of MERIT were modified, in accordance with the decisions described above. Of course, the mpeg2stat files were modified. First of all, all *Trace* and *Verbose* outputs were commented out. There is

no test to see whether `Trace_Flag` or `Verbose_Flag` is on or off, to increase processing speed. Moreover, all parts involving an enhancement layer were commented out, since MERIT uses only the base layer file. The following files were modified:

- `global.h`

  `Global.h` and `List.h` were included from MERIT, as well as all needed external variables and functions (mainly used in the *InfoList building algorithm*)

- `mpeg2dec.c`

  The `main()` function was renamed `mpeg_stat()` (not `mpeg2stat()`, so there was no need to change the name when calling in files from MERIT). There is a short options setup. The *InfoList building* starts here with `FRAME Line` update, and a call to *initInfo()*. Obsolete functions such as `Process_options()`, `Usage()`, `Print_options()` and `Output_Statistics()` have been cut off.

- `gethdr.c`

  *InfoList building* for `SLICE Line` and `GOP Line`. The lines required by this algorithm were added, including reset of the `current` pointer, update of the pointed `Line` structure with the parsed information, and calls to the *initInfo()* functions.

- `getpic.c`

  *InfoList building* for `BLOCK Line` as explained for `SLICE` and `GOP`, with some needed computations such as motion vector(s). Moreover, when dealing with a field picture, a `Line` buffer is built, so that one macroblock can be sent twice to MERIT via the `InfoList`.

- `getblk.c`

  Test for `dct` flag. If `dct` = 1, DCT coefficients are stored in a `dctSpecifics` string. The coefficients are stored in this string on the fly while parsing the stream. This test improves processing speed when blockinfo `mpeg_mode` is used.

The other files from mpeg2stat have not been modified and are used as in mpeg2stat.

### 4.4 Tests and further possible improvements

This version was tested with several files. The results were as follows:

- For MPEG-1 files, the results are exactly the same, for all possible options. This was predictable, since only the parsing part was modified. The only difference is the processing speed which is sensibly lower with this new version. This is probably due to the increased number of tests during the parsing process.

- For MPEG-2 files using *frame pictures* (*frame* or *field prediction*, *frame* or *field dct*) the results are similar to those for MPEG-1 files. All the cuts are found for unambiguous clips (as in MPEG-1). The DCT based segmentation algorithm

gives odd results, but this already occurred with MERIT 3.3. The `-showKey` and similar options do not work with MPEG-2 files, because these options directly use `mpeg_play`, which doesn't support MPEG-2 files. The options used by `mpeg_play` in this case are not provided by `mpeg2play`.

- For MPEG-2 files with *field pictures*, the results are good as well. Of course, with the `-dispMV` option, two consecutive lines of macroblocks are identical, and the displayed macroblock types do not show the reality of the encoded picture. But this solution seems to work. The segmentation is actually performed on field pictures with half the height of displayed frame pictures.

All the test files were in 4:2:0 chroma format. MERIT 4.0 has not yet been tested with other formats, but the block-based segmentation should also work for those. For the DCT segmentation, MERIT currently stops the parsing of the `dctSpecifics` string after the 6th block, so that only the first two chrominance blocks are taken into account.

It is worth noting that all the tests were performed with reconstructed MPEG-2 files. That is, it was necessary to decode MPEG-1 files and encode them as MPEG-2 files, because MPEG-2 video sequences with cuts could not be found on the Internet. However, the MSSG encoder mpeg2encode was used, and it was possible to construct almost all types of MPEG-2 files. For interlaced video, interlaced pictures decoded from an MPEG-2 video clip without cuts were used, and they were reordered to provide some cuts in the clip before encoding. It was not possible to test *Dual Prime* predicted MBs nor clips with both *field-* and *frame-pictures* in the same stream, because mpeg2encode does not provide for these possibilities.

Here is a non-exhaustive list of possible improvements:

- A solution should be found for the `-showKey` option. Possibly more options should be added to mpeg2play.
- For sequences with *field-pictures*, an efficient use of the second field of a frame could be added, at least to provide a correct output when using the `-dispMV` option.

## 5 Conclusion

The new version of MERIT is now able to analyze MPEG-2 files. The modifications were made in accordance with the MPEG-2 specifications and using the original segmentation algorithm. MPEG-1 files are analyzed exactly as in the former version, and in case of MPEG-2 files, the information is processed before being analyzed by the segmentation algorithm. This approach gives good results, according to our tests.

# A Appendix: The MPEG-1 and MPEG-2 video standards

## A.1 Introduction

The Moving Picture Experts Group (MPEG) [12] is a working group of ISO/IEC in charge of the development of international standards for compression, decompression, processing, and code representation of moving pictures, audio and their combination. The MPEG-1 standard was approved in Nov. 1992, and MPEG-2 in Nov. 1994.

## A.2 MPEG video coder source model

The MPEG digital video coding techniques are statistical in nature [7]. Video sequences usually contain statistical redundancies in both temporal and spatial dimensions. The basic statistical property upon which MPEG compression techniques rely is inter-pixel correlation, including the assumption of simple correlated translational motion between consecutive frames. Thus, it is assumed that the magnitude of a particular image pixel can be predicted from nearby pixels within the same frame (using Intra-frame coding techniques) or from pixels of a nearby frame (using Inter-frame techniques). The MPEG compression algorithms employ Discrete Cosine Transform (DCT) coding techniques on image blocks of $8 \times 8$ pixels to efficiently exploit spatial correlations between nearby pixels within the same image.

However, if the correlation between pixels in nearby frames is high, i.e. in cases where two consecutive frames have similar or identical content, it is desirable to use Inter-frame DPCM coding techniques employing temporal prediction (motion-compensated prediction between frames). In MPEG video coding schemes an adaptive combination of both temporal motion-compensated prediction followed by transform coding of the remaining spatial information is used to achieve high data compression (hybrid DPCM/DCT coding of video).

## A.3 Compression techniques

### A.3.1 Subsampling and interpolation

The basic concept of subsampling is to reduce the dimension of the input video (horizontal dimension and/or vertical dimension) and thus the number of pixels to be coded prior to the encoding process. At the receiver the decoded images are interpolated for display. Since the human eye is more sensitive to changes in brightness than to chromaticity changes, the MPEG coding schemes first divide the images into YUV components (one luminance and two chrominance components); then, the chrominance components are subsampled relative to the luminance component with a Y:U:V ratio specific to particular applications (with the MPEG-2 standard, a ratio of 4:1:1, 4:2:2, or 4:4:4 is used).

### A.3.2 Transform domain coding

The purpose of transform coding is to de-correlate the image content and to encode transform coefficients rather than the original pixels of the images. To this end the input images are split into disjoint blocks of pixels. Of many possible alternatives, the *Discrete*

*Cosine Transform* (DCT) applied to small image blocks, usually $8 \times 8$ pixels, has become the most successful transform for still image and video coding. A major objective of transform coding is to make as many transform coefficients as possible small enough so that they are insignificant and need not be coded for transmission.

The DCT coefficients are put into a matrix, called the DCT matrix. On average only a small number of DCT coefficients need to be transmitted to the receiver to obtain a valuable approximate reconstruction of the image blocks. Moreover, the most significant DCT coefficients are concentrated around the upper left corner of the matrix (low DCT coefficients) and the significance of the coefficients decays with increased distance. Since the human viewer is more sensitive to reconstruction errors related to low spatial frequencies than to high frequencies, a frequency-adaptive weighting (quantization) of the coefficients according to human visual perception (perceptual quantization) is often employed to improve the visual quality of the decoded images for a given bit rate.

### A.3.3 Motion compensated prediction

The concept of motion compensation is based on the estimation of motion between video frames, i.e. if all elements in a video scene are spatially displaced, the motion between frames can be described by a motion vector. To this end images are usually separated into disjoint blocks of pixels ($16 \times 16$ pixels in the MPEG-1 and MPEG-2 standards) and only one motion vector is estimated, coded and transmitted for each of these blocks. In the MPEG compression algorithms the motion-compensated prediction techniques are used for reducing temporal redundancies between frames and only the prediction error images — the differences between the original images and the motion compensated prediction images — are encoded, using the DCT technique.

## A.4 The MPEG-1 standard

The MPEG-1 standard is formally referred to as ISO 11172 and consists of several parts (1. System, 2. Video, 3. Audio, 4. Conformance, 5. Software). We will focus on the video part 11172-2. The MPEG-1 Video standard was originally aimed at coding video of SIF resolution ($352 \times 240$ at 30 noninterlaced frames/s or $352 \times 288$ at 25 noninterlaced frames/s) at bit rates of about 1.5 Mbits/s, for applications such as CD-i (compact disc interactive). However, it also allows much larger picture sizes and correspondingly higher bit rates. This standard specifies the video bit stream syntax and the corresponding video decoding process. The basic MPEG-1 video compression technique is based on a macroblock structure, motion compensation, and the conditional replenishment of macroblocks.

### A.4.1 Macroblocks and I-frames

The MPEG-1 coding algorithm encodes the first frame of a video sequence in *Intra-frame* coding mode (*I-picture*), by using block-based DCT coding of $8 \times 8$ pixel blocks, followed by quantization. The DC coefficient (average value and first coefficient) is quantized with a uniform midstep quantizer with stepsize as specified by the parameter `intra_dc_precision` $= 3 - \log_2$ *stepsize*. AC coefficients (the other 61 coefficients)

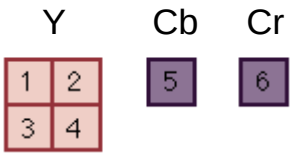


Figure 1: Macroblock structure [6]. A macroblock comprises four luminance blocks (Y1, Y2, Y3, Y4) and two chrominance blocks (Cr, Cb). Each block has a size of $8 \times 8$ pixels.

are quantized with a uniform midstep quantizer having a stepsize under control of the parameter `quantizer_scale`. A high stepsize decreases the number of bits needed to transmit the information, but also decreases the image quality. Each color input frame in a video sequence is partitioned into non-overlapping *macroblocks*. Each macroblock contains six $8 \times 8$ blocks of data from both luminance and co-sited chrominance bands — four luminance blocks and two chrominance blocks, each of size $8 \times 8$ pixels (Fig. 1). Thus the sampling ratio between Y:U:V luminance and chrominance pixels is 4:1:1 (sometimes called 4:2:0). For an I-picture, the frame is partitioned into such macroblocks. Then each luminance and chrominance block from each macroblock is coded using the DCT technique.

### A.4.2 Zig-zag scanning

The concept of *zig-zag scanning* of the coefficients is outlined in Fig. 2. The zig-zag scan attempts to trace the DCT coefficients according to their significance, from the top-left corner to the bottom-right corner. Only the non-zero quantized DCT coefficients are encoded. The scanning of the quantized DCT-domain 2-dimensional signal followed by variable-length code-word assignment (*Huffman coding*) for the coefficients serves as a mapping of the 2-dimensional image signal into a 1-dimensional bit stream. The non-zero AC coefficient quantizer values (*length*) are detected along the scan line as well as the distance (*run*) between two consecutive non-zero coefficients. Each consecutive (run, length) pair is encoded by transmitting only one codeword (*Run Length Encoding*).

### A.4.3 P-frames and B-frames

Each subsequent frame is coded using *Inter-frame prediction* (predicted pictures, or *P-pictures*) — only data from the nearest previously coded I- or P-frame is used for prediction. For coding P-pictures, the previous I- or P-picture frame $N-1$ is stored in a frame store in both encoder and decoder. Motion compensation is performed on a macroblock basis — only one motion vector is estimated between frame $N$ and frame $N-1$ for a particular macroblock to be encoded. These motion vectors are then coded. The motion-compensated prediction error is calculated by subtracting each pixel in a macroblock from its motion-shifted counterpart in the previous frame. A $8 \times 8$ DCT is then applied to each of the $8 \times 8$ blocks contained in the error-image-macroblock, followed by quantization of the DCT coefficients with subsequent zig-zag scanning and

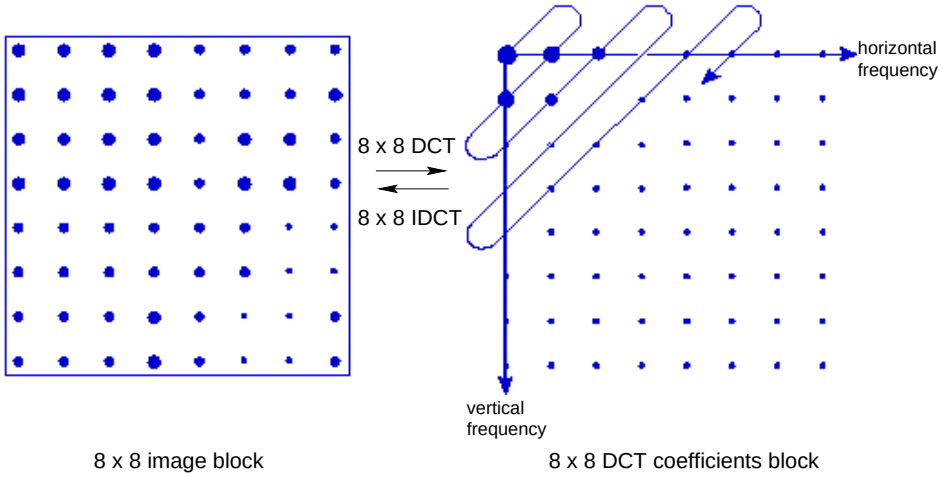


Figure 2: Zig-zag scan of the DCT coefficients of an $8 \times 8$ block. [8]

run-length coding. The quantization stepsize can be adjusted for each macroblock in a frame (Fig. 3). The advantage of coding video using motion compensation techniques is the reduction of the residual signal to be coded compared to pure frame difference coding.

To further explore the significant advantages of motion compensation and motion interpolation, the concept of *B-pictures* (bidirectionally predicted pictures) was introduced by MPEG-1. B-pictures can be coded using motion-compensated prediction based on the two nearest already coded frames (either I-pictures or P-pictures). Since the coding order of the pictures is not the same as the displaying order, B-pictures can use both past and future frames as references (Fig. 4). The user can arrange the picture types in a video sequence with a high degree of flexibility to suit diverse application requirements.

### A.4.4 Conditional replenishment

An essential feature supported by MPEG-1 is the possibility of updating macroblock information at the decoder only if needed (i.e. if the content of the macroblock has changed in comparison to the content of the same macroblock in the previous frame). This feature is called *conditional replenishment*. The key to efficient coding of video sequences at low bit rates is the selection of appropriate prediction modes to achieve conditional replenishment. The MPEG-1 standard distinguishes between three different macroblock coding types (MB types):

Skipped MB — prediction from previous frame with zero motion vector. No information about the macroblock is coded or transmitted to the receiver.

Inter MB — motion-compensated prediction from the previous frame is used. The MB type, the MB address and, if required, the motion vector, the DCT coefficients and quantization stepsize are transmitted.

Intra MB — no prediction is used from the previous frame (intra-frame prediction only). Only the MB type, the MB address, the DCT coefficients and the quantization

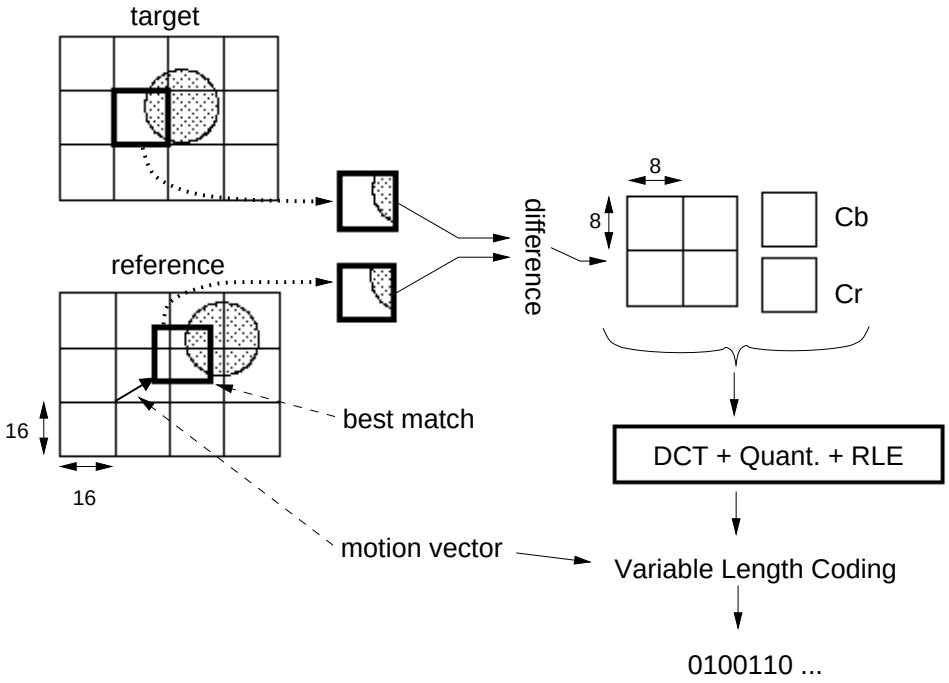


Figure 3: Forward prediction of a macroblock in a P-picture [6]. DCT: Discrete Cosine Transform, Quant.: Quantization, RLE: Run Length Encoding

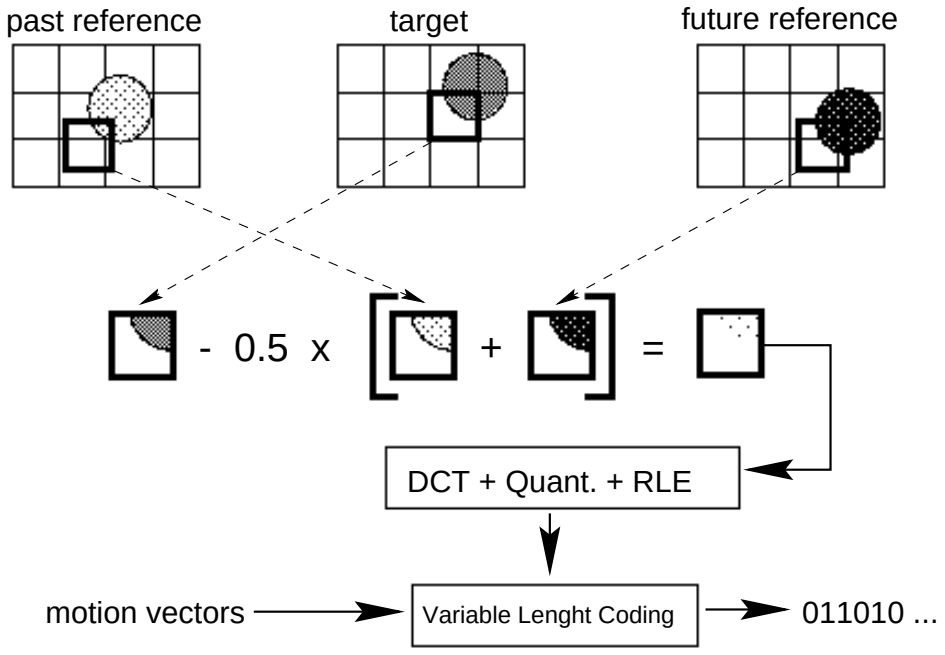


Figure 4: Bidirectional prediction of a macroblock in a B-picture [6]. DCT: Discrete Cosine Transform, Quant.: Quantization, RLE: Run Length Encoding

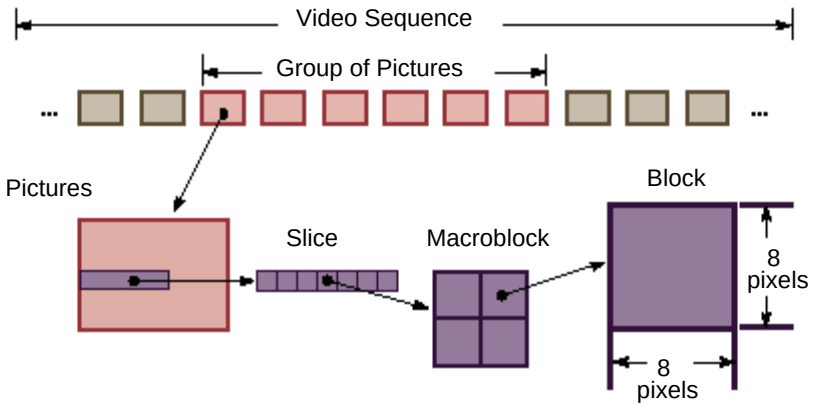


Figure 5: Video sequence structure as specified by MPEG [6]. A Group of Pictures usually begins with an I-picture, followed by P- and B-pictures. Each picture is divided into slices, macroblocks and blocks.

stepsize are transmitted to the receiver.

### A.4.5 The MPEG-1 video stream syntax

Here is a coarse view of the video stream syntax. In typical MPEG-1 encoding, an input video sequence is divided into units of *groups-of-pictures* (GOPs), where each GOP consists of an arrangement of one I-picture, P-pictures, and B-pictures. A GOP serves as a basic access unit. Each picture is divided further into one or more *slices* that offer a mechanism for resynchronization and thus limit the propagation of errors. Each slice is composed of a number of *macroblocks*. Each macroblock is composed of four $8 \times 8$ luminance blocks and two chrominance blocks (see Fig. 5). In P-pictures each macroblock can have one motion vector, whereas in B-pictures each macroblock can have as many as two motion vectors. Each part of this syntax (GOPs, Pictures, Slices) is encoded with a special header (as shown in Fig. 6), and programs such as decoders or MERIT parse the MPEG-file video bit stream, and recognize these headers. Thus, they can efficiently deal with further data.

## A.5 The MPEG-2 standard

MPEG-2 is an extension of the MPEG-1 international standard for digital compression of audio and video signals. MPEG-2 is directed at broadcast formats at higher data rates; it provides extra algorithmic tools for efficiently coding interlaced video, supports a wide range of bit rates, and provides for multichannel surround sound coding. It is also known as ISO/IEC 13818.

The MPEG-2 standard is capable of coding standard-definition television at bit rates from about 3–15 Mbit/s and high-definition television at 15–30 Mbit/s. Since MPEG-2 is a superset of MPEG-1, MPEG-2 decoders will also decode MPEG-1 bit streams.

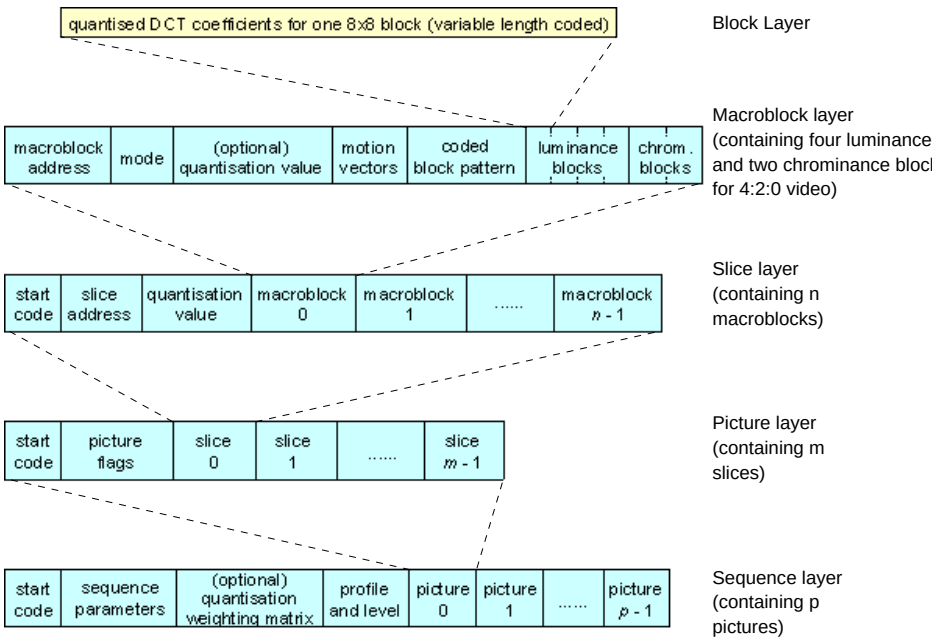


Figure 6: Bit stream structure as specified by MPEG [8]. Each picture is divided into $m$ horizontal slices, each comprising $n$ macroblocks. For 4:2:0 video, each macroblock contains four luminance and two chrominance $8 \times 8$ blocks of quantized DCT coefficients. The profile and level indication appears only in MPEG-2 files.

### A.5.1 Profiles and levels

The implementation of the full syntax of MPEG-2 may not be practical for most applications. MPEG-2 has introduced the concept of "Profiles" and "Levels" to stipulate conformance between equipment not supporting the full implementation. Profiles and levels provide means for defining subsets of the syntax and thus the decoder capabilities required to decode a particular bit stream. A profile is a subset of algorithmic tools and a level identifies a set of constraints on parameter values (such as picture size and bit rate). The specifications of each level and each profile are described in Table 1 and Table 2. A decoder that supports a particular profile and level is only required to support the corresponding subset of the full standard and set of parameter constraints. The main profile at the main level is the most common type and is referred to as *MP@ML*.

Currently, the major interest is in the main profile at the main level for applications such as digital television broadcasting (terrestrial, satellite and cable), video-on-demand services, and desktop video systems.

### A.5.2 MPEG-2 MAIN Profile and MPEG-1

The MPEG-2 algorithm defined in the MAIN Profile is a straightforward extension of the MPEG-1 coding scheme to accommodate coding of interlaced video, while retaining the full range of functionality provided by MPEG-1. Identical to the MPEG-1 standard, the MPEG-2 coding algorithm is based on the general hybrid DCT/DPCM coding

| Level | Parameters |
|---|---|
| High | 1920 samples/line<br>1152 lines/frame<br>60 frames/s<br>80 Mbit/s |
| High 1440 | 1440 samples/line<br>1152 lines/frame<br>60 frames/s<br>60 Mbit/s |
| Main | 720 samples/line<br>576 lines/frame<br>30 frames/s<br>15 Mbit/s |
| Low | 352 samples/line<br>288 lines/frame<br>30 frames/s<br>4 Mbit/s |

Table 1: Upper bounds of parameters at each level of a profile.

| Profile | Algorithms |
|---|---|
| High | Supports all functionality provided by the Spatial Scalable Profile plus the provision to support three layers with the SNR and Spatial scalable coding modes. 4:2:2 YUV-representation for improved quality requirements |
| Spatial scalable | Supports all functionality provided by the SNR Scalable Profile plus an algorithm for Spatial scalable coding (two layers allowed). 4:0:0 YUV-representation |
| SNR scalable | Supports all functionality provided by the Main Profile plus an algorithm for SNR scalable coding (two layers allowed). 4:2:0 YUV-representation |
| Main | Non-scalable coding algorithm supporting functionality for coding interlaced video, random access and B-picture prediction modes. 4:2:0 YUV-representation |
| Simple | Includes all functionality provided by the Main Profile but does not support B-picture prediction modes. 4:2:0 YUV-representation |

Table 2: Algorithms and functionalities supported by each profile.

scheme, incorporating a macroblock structure, motion compensation, and coding modes for conditional replenishment of macroblocks. The concept of I-pictures, P-pictures and B-pictures is fully retained in MPEG-2 to achieve efficient motion prediction and to assist random-access functionality.

### A.5.3 Interlaced video

The MPEG-1 standard deals only with progressive video. That is, all the pixels of a frame have been taken at the same instant, as in film. The original objective of MPEG-2 was to efficiently code interlaced video, which is mainly used in television. Television services in the United States currently broadcast video at a frame rate of just under 30 Hz (29.97 Hz). Each frame consists of two interlaced fields, giving a field rate of approximately 60 Hz. The first field of each frame contains only the odd-numbered lines (*top field*) of the frame (numbering the top frame line as line 1). The second field contains only the even-numbered lines (*bottom field*) of the frame and is sampled in the video camera 20 ms after the first field. It is important to note that one interlaced frame contains fields from two instants in time. European television is similarly interlaced but with a frame rate of 25 Hz.

MPEG-2 introduced the concept of frame pictures and field pictures along with particular frame prediction and field prediction modes to accommodate coding of progressive and interlaced video. For interlaced sequences it is assumed that the coder input consists of a series of odd and even fields that are separated in time by a field period. Two fields of a frame may be coded separately (field pictures). In this case each field is separated into adjacent non-overlapping macroblocks and the DCT is applied on a field basis. Alternatively, two fields may be coded together as a frame (frame pictures), as in conventional coding of progressive video sequences. Here, consecutive lines of the top and bottom fields are simply merged to form a frame. It is worth noting that both frame pictures and field pictures can be used in a single video sequence.

### A.5.4 Picture types

The MPEG-2 syntax specifies the different types of pictures that may be coded and displayed [9]. Several variables are used to define a picture type, as shown in Fig. 7. Here we will focus only on the semantic meaning of the relevant variables.

- Sequence level:

  `progressive_sequence` is a 1-bit integer, which indicates whether the sequence contains only progressive frame pictures (1) or not (0). This is the primary switch between interlaced and progressive video sources, and gives the display mode (progressive or interlaced)

- Picture level:

  `progressive_frame` (1-bit integer) 1 indicates that the two fields of the frame correspond to the same instant, for example, film. It gives the video caption

mode (progressive or interlaced video). If `progressive_sequence` = 1 then we must have `progressive_frame` = 1. That is, an originally interlaced frame cannot be displayed as a progressive frame. Nevertheless, MPEG-2 allows for progressive coded pictures, but interlaced display (`frame_picture` = 1 and `progressive_sequence` = 0).

`picture_structure` defines the way a picture is internally coded. This 2-bit integer takes the following values: 3 (11) for a frame picture, 1 (01) for a top-field picture, and 2 (10) for a bottom-field picture. In the case of frame pictures, the two fields are interleaved to form a frame, and for field pictures, the two fields are coded separately. In this case, the picture coding type (I, P, B) is the same for the two fields of the same frame, except for I-pictures, where the second field can be a P-picture. If `progressive_frame` = 1 (progressive coded picture), the picture must have a frame structure.

`top_field_first` and `repeat_first_field` are two indicators whose meaning depends on the values of `progressive_sequence` and `picture_structure`. If `progressive_sequence` = 0 and `picture_structure` = 3 (frame), then `top_field_first` indicates which of the two fields must be displayed first (top:1 bottom:0) and `repeat_first_field` indicates whether the first field should be repeated to respect the top-bottom display. Note that `repeat_first_field` cannot be equal to 1 if `progressive_frame` = 0. If `picture_structure` = 1 or 2 (field pictures), then `top_field_first` = 0 and `repeat_first_field` = 0, and the display order is first coded first displayed. If `progressive_sequence` = 1 then the two bits `[top_field_first, repeat_first_field]` give the number of times the progressive picture is displayed (00: 1 frame (MP@ML), 01: 2 frames, 11: 3 frames)

To summarize, `progressive_sequence` gives the display mode (interlaced/progressive), `progressive_frame` the original type of picture (interlaced/progressive), and `picture_structure` the picture storage mode (frame picture/field picture).

### A.5.5 Frame and field predictions

New motion-compensated field prediction modes were introduced by MPEG-2 to efficiently encode field pictures and frame pictures. In field prediction, predictions are made independently for each field by using data from one or more previously decoded fields, i.e. for a top field a prediction may be obtained from either a previously decoded top field (using motion compensated prediction) or from the previously decoded bottom field belonging to the same picture. An indication of which reference field is used for prediction is transmitted with the bit stream. Within a field picture all predictions are field predictions.

Frame prediction makes a prediction for a frame picture based on one or more previously decoded frames. In a frame picture either field or frame predictions may be used and the particular prediction mode preferred can be selected on a macroblock-by-macroblock basis. Here is a description of each possible prediction type.

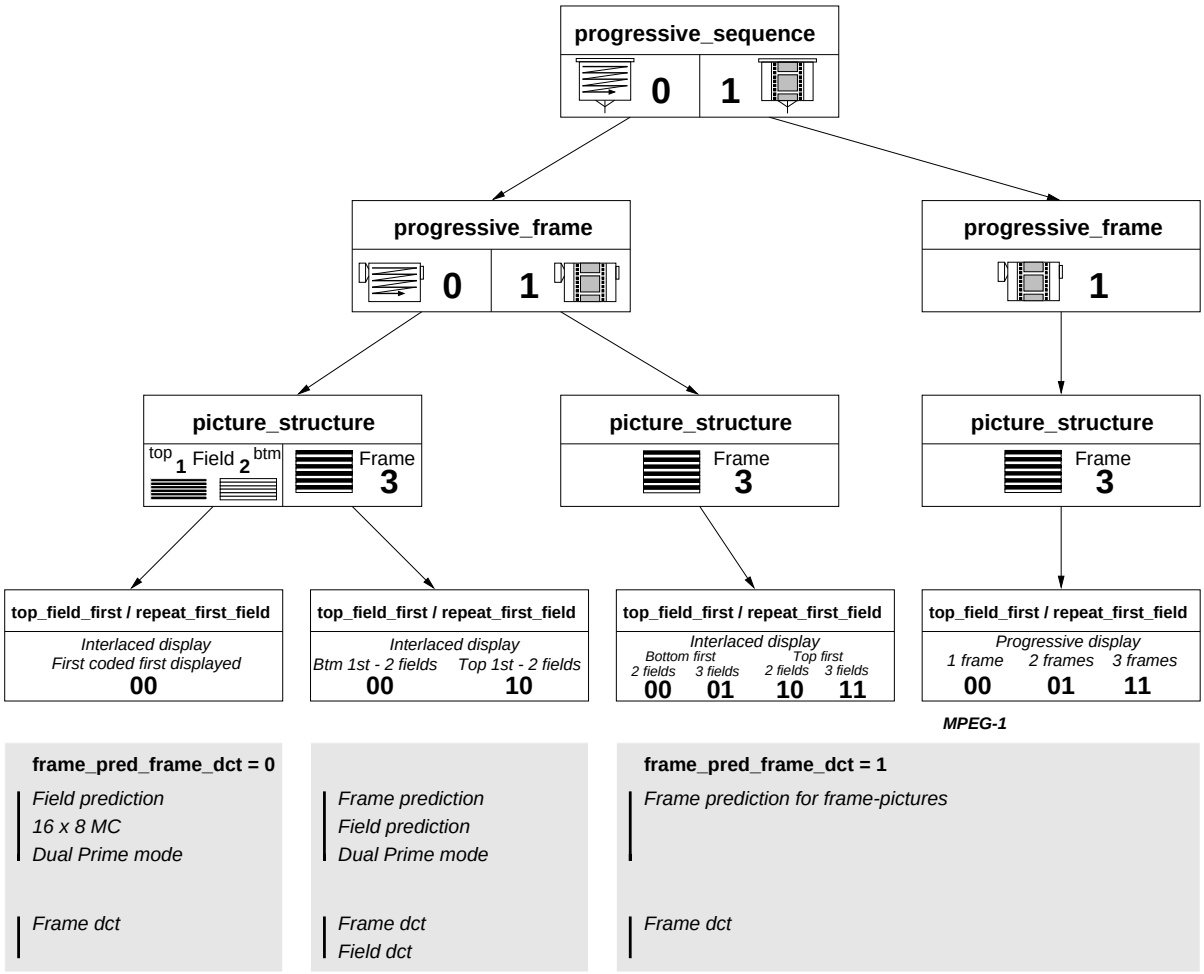


Figure 7: MPEG-2 picture types and corresponding prediction modes and dct types.

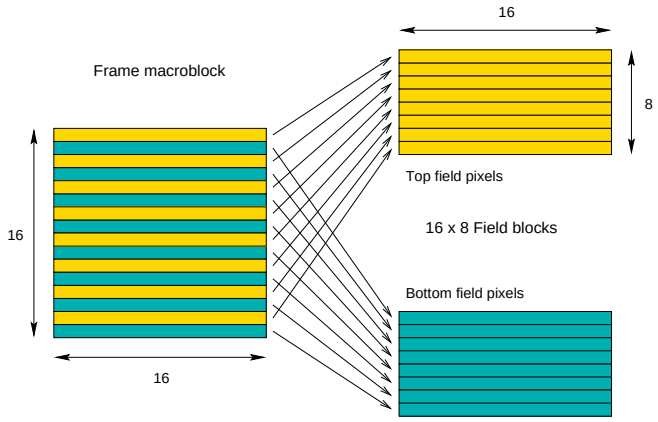


Figure 8: Field prediction for frame pictures. Target macroblocks are split into top-field pixels and bottom-field pixels.

- Frame pictures

  **Frame prediction** is exactly the same as in MPEG-1. Each macroblock has up to one motion vector (forward prediction) in P-frames and up to two motion vectors (forward and backward) in B-frames.

  **Field prediction for frame pictures** is a prediction mode where the target macroblock is first split into top-field pixels and bottom-field pixels, constituting two $16 \times 8$ "field macroblocks" (see Fig. 8). Then, for each of these two half-MBs, the prediction half-MB(s) is (are) found in previous reference fields. For P-frame pictures, the prediction half-MB may come from either field of the two most recently coded I- or P-frames. For B-frame pictures, the backward prediction half-MB is taken from either field of the most recently coded I- or P-frame, and the forward prediction half-MB from either field of the last but one I- or P-frame. To select the prediction field used, MPEG fills the `motion_vertical_field_select[r][s]` variable as described below. Up to two motion vectors are assigned to each MB in a P-frame picture (one for each half-field-MB), and up to four in a B-frame picture.

  `motion_vertical_field_select[r][s]` gives the field selected for the prediction. Index r=0 indicates the first MV (first half-MB) and r=1 indicates the second MV (second half-MB). Index s=0 indicates the forward MV, s=1 the backward MV. 0 indicates the top field, 1 indicates the bottom field.

  **Dual-Prime mode** is only used for P-pictures. With this mode there is only one motion vector, from which two preliminary predictions are computed. The first preliminary prediction is identical to frame prediction, except that each prediction pixel must have the same parity as the target pixel. The second preliminary prediction is derived using a computed motion vector plus a small differential motion vector (*dmvector*). The computed motion vector is obtained by a temporal scaling of the transmitted motion vector, and for the

final corrected motion vector, each prediction pixel has opposite parity to the target pixel. The two preliminary predictions are then averaged together to form the final prediction.

- Field pictures

  **Field prediction for field pictures** is similar to field prediction for frame pictures, except that there is no half-MB (in field pictures all the pixels belong to the same field). So for a particular MB, all pixels have the same parity (i.e. they all come from the same field). For P-field pictures, the prediction MB may come from either of the two most recently coded I- or P-fields, even when coding the second field of a frame, if the prediction field is the first field of the same frame. For B-field pictures, the backward prediction MB is taken from either field of the most recently coded I- or P-frame, and the forward prediction MB from either field of the last but one I- or P-frame. For field selection, the motion_vertical_field_select[r][s] variable is also used, but with r=0 only (only one MB). Up to one motion vector is assigned to each MB in a P-field picture and up to two in a B-field picture.

  $16 \times 8$ **MC prediction mode** is similar to field prediction for frame pictures: The target MB is split into an upper half and a lower half, and a separate field prediction is performed for each half, as in field prediction for frame pictures. Up to two motion vectors are assigned to each MB in a P-frame picture (one for each half-field-MB), and up to four in a B-frame picture.

  **Dual-Prime mode** for P-pictures is the same as in frame pictures except that for the two preliminary predictions, reference pixels are taken from only one field: the same parity as the target MB for the first prediction, and the opposite parity for the second prediction.

There are three motion modes for each picture type (frame or field), one using one motion vector for P-pictures and two for B-pictures (frame prediction for frame pictures, field prediction for field pictures), another using two motion vectors for P-pictures and four for B-pictures (field-prediction for frame pictures, $16 \times 8$ MC for field pictures), and the last using one motion vector and a dmvector (Dual-Prime for P-pictures). It is worth noting that the macroblock type seems to be chosen before the motion compensation mode in most encoders.

### A.5.6 Frame-dct and field-dct coding

MPEG-2 provides another feature for dealing with interlaced pictures. For frame pictures, on a macroblock-by-macroblock basis, the dct_type can be set as frame_dct (0) or field_dct (1). The frame dct type is the same dct coding as in MPEG-1. With the field dct type, just prior to performing the DCT, the encoder may reorder the luminance lines within a MB so that the first 8 lines come from the top field, and the last 8 lines come from the bottom field. This reordering is undertaken just after the Inverse DCT. With field_dct

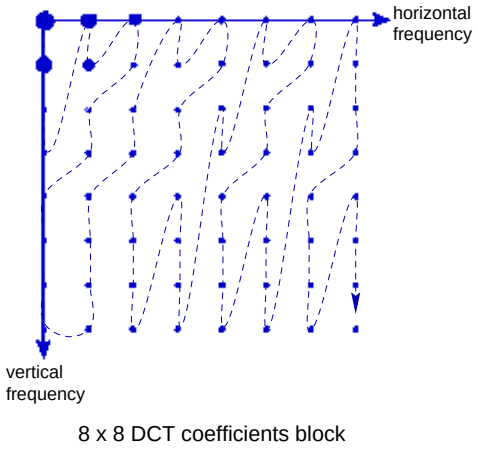


Figure 9: Alternate scan of the DCT coefficients of an $8 \times 8$ block.

in an interlaced frame picture, the vertical correlation within the luminance blocks is increased.

For each frame, the `frame_pred_frame_dct` is a kind of shortcut variable. If `frame_pred_frame_dct` = 1, then only frame prediction and frame DCT are used within the frame. If `frame_pred_frame_dct` = 0 then all the motion compensation modes and all the DCT types can be used. If `progressive_frame` = 1 then `frame_pred_frame_dct` = 1, and if `picture_structure` = 1 or 2, then `frame_pred_frame_dct` = 0.

In fact, for each picture type (as described before), only specific types of compensation modes and dct coding are allowed (see also Fig. 7).

### A.5.7 Alternate scan

The main effect of interlace in frame pictures is that since adjacent scan lines come from different fields, vertical correlation is reduced when there is motion in the scene. This vertical correlation reduction provides a non-optimum zig-zag scanning order. That is the reason why MPEG-2 has an Alternate-Scan mode, shown in Fig. 9. The type of scan may be specified by the encoder on a picture-by-picture basis.

### A.5.8 Chrominance formats

MPEG-2 specifies additional Y:U:V luminance and chrominance subsampling ratio formats to assist and foster applications with high video quality requirements. In addition to the 4:1:1 format already supported by MPEG-1 the specification of MPEG-2 is extended to 4:2:2 and 4:4:4 formats suitable for studio video coding applications. 4:2:2 means the chrominance is horizontally subsampled by a factor of two relative to the luminance; 4:1:1 (also called 4:2:0) means the chrominance is horizontally and vertically subsampled by a factor of two relative to the luminance. In the MAIN Profile at MAIN Level, only the 4:2:0 format is allowed.

### A.5.9 MPEG-2 scalability techniques

The scalability tools standardized by MPEG-2 support applications beyond those addressed by the basic MAIN Profile coding algorithm. The intention of scalable coding is to provide interoperability between different services and to flexibly support receivers with different display capabilities. Receivers either unable or unwilling to reconstruct the full-resolution video can decode subsets of the layered bit stream to display video at lower spatial or temporal resolution or with lower quality. Another important purpose of scalable coding is to provide a layered video bit stream which is amenable to prioritized transmission.

For instance, two layers can be provided, each layer supporting video at a different scale, i.e. a multiresolution representation can be achieved by downscaling the input video signal into a lower-resolution video (downsampling spatially or temporally). The downscaled version is encoded into a base layer bit stream with a reduced bit rate. The upscaled reconstructed base layer video (upsampled spatially or temporally) is used as a prediction for the coding of the original input video signal. The prediction error is encoded into an enhancement layer bit stream. If a receiver is either unable or unwilling to display the full-quality video, a downscaled video signal can be reconstructed by decoding only the base layer bit stream. Thus scalable coding can be used to encode video with a suitable bit rate allocated to each layer in order to meet specific bandwidth requirements of transmission channels or storage media. Browsing through video data bases and transmission of video over heterogeneous networks are applications expected to benefit from this functionality.

### A.5.10 The MPEG-2 video stream syntax

The MPEG-2 video standard specifies the syntax and semantics of the compressed video stream produced by the video encoder. Most of MPEG-2 consists of additions to MPEG-1. The video stream syntax is flexible to support the variety of applications envisaged for the MPEG-2 video standard. Like MPEG-1, the syntax is constructed in a hierarchy of headers which are: Video sequence header, Group of Pictures header, Picture header, Slice header and Macroblock header. The block contains the DCT coefficients (see Fig. 5 and Fig. 6). Useful information about these headers can be found in the MPEG specifications [9].

## B Appendix: UML diagrams

These UML diagrams (Figs. 10 to 13) provide further explanations about the structure of MERIT 4.0. Since MERIT is not an object-oriented application, class diagrams are not used here. The structure diagrams show the structure of MERIT and of typical MPEG files, and the collaboration diagrams depict the interactions between the different components of MERIT.

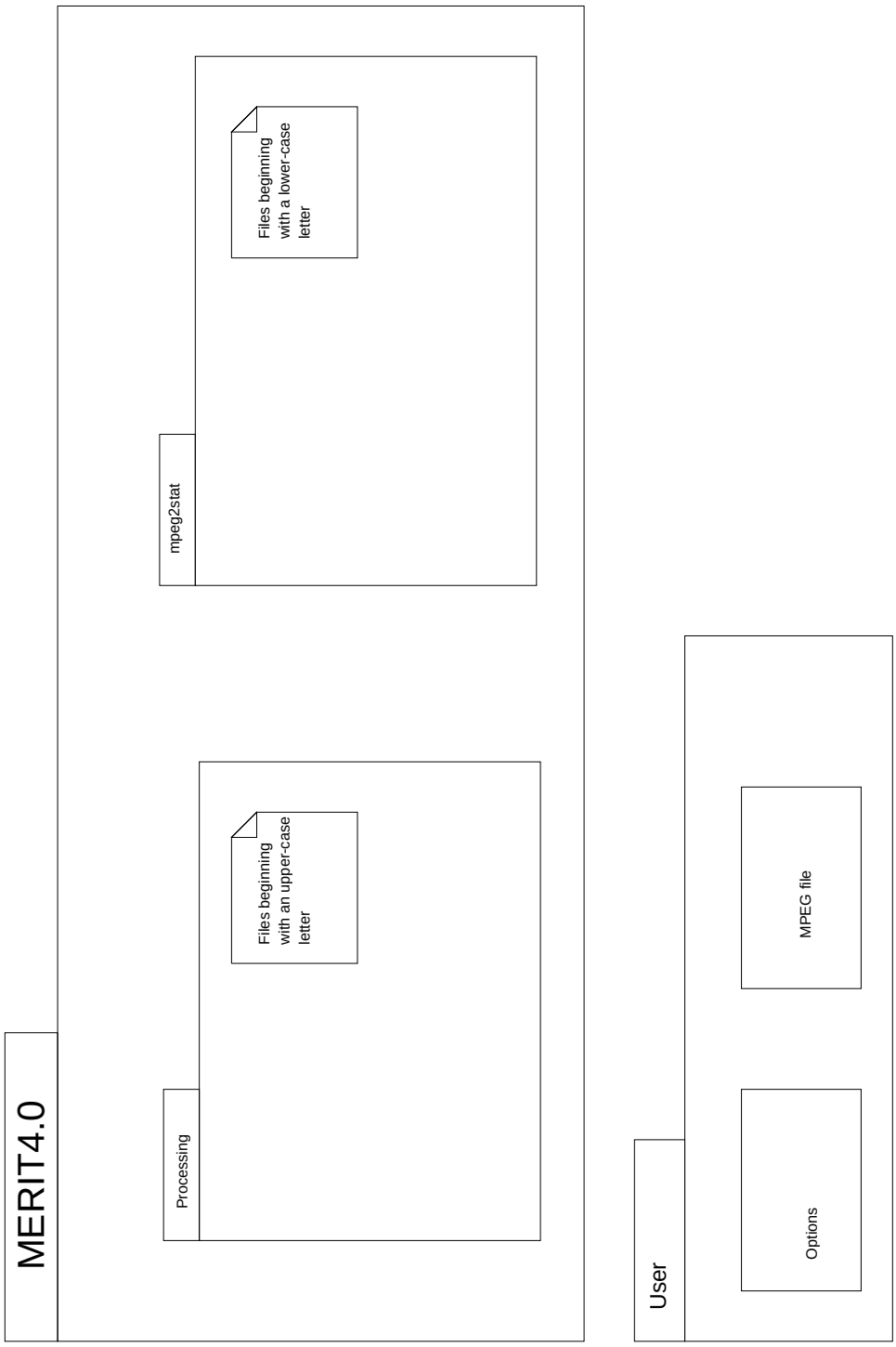


Figure 10: MERIT 4.0 general structure diagram.

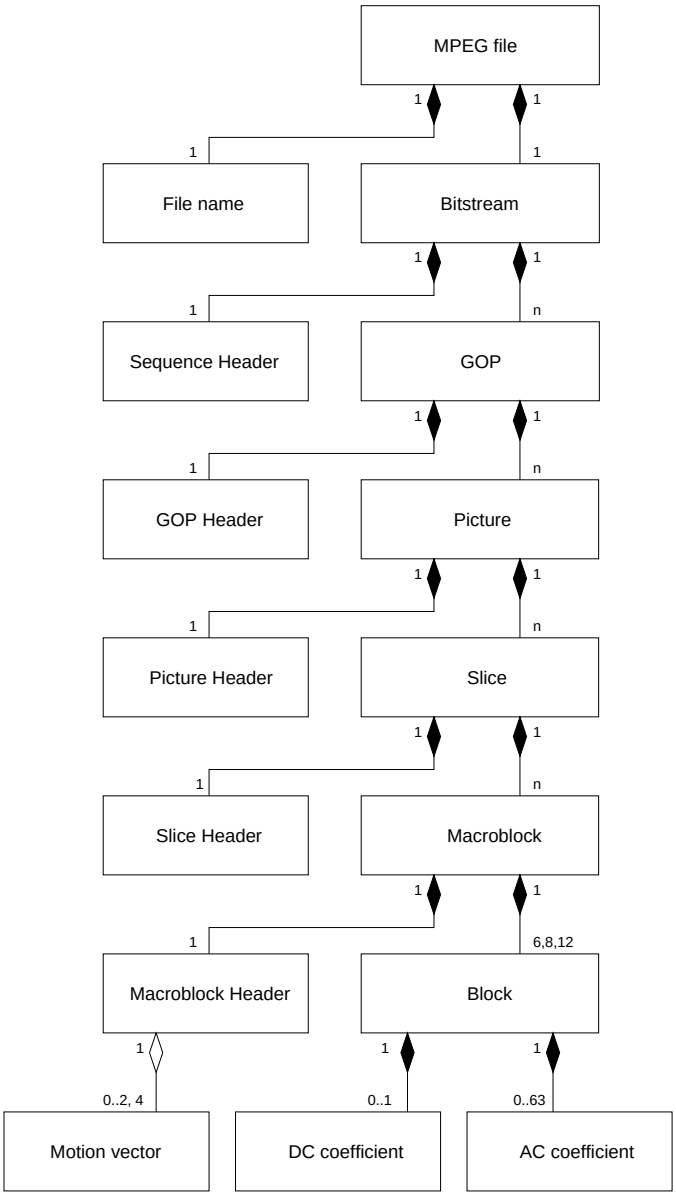


Figure 11: MPEG file structure diagram.

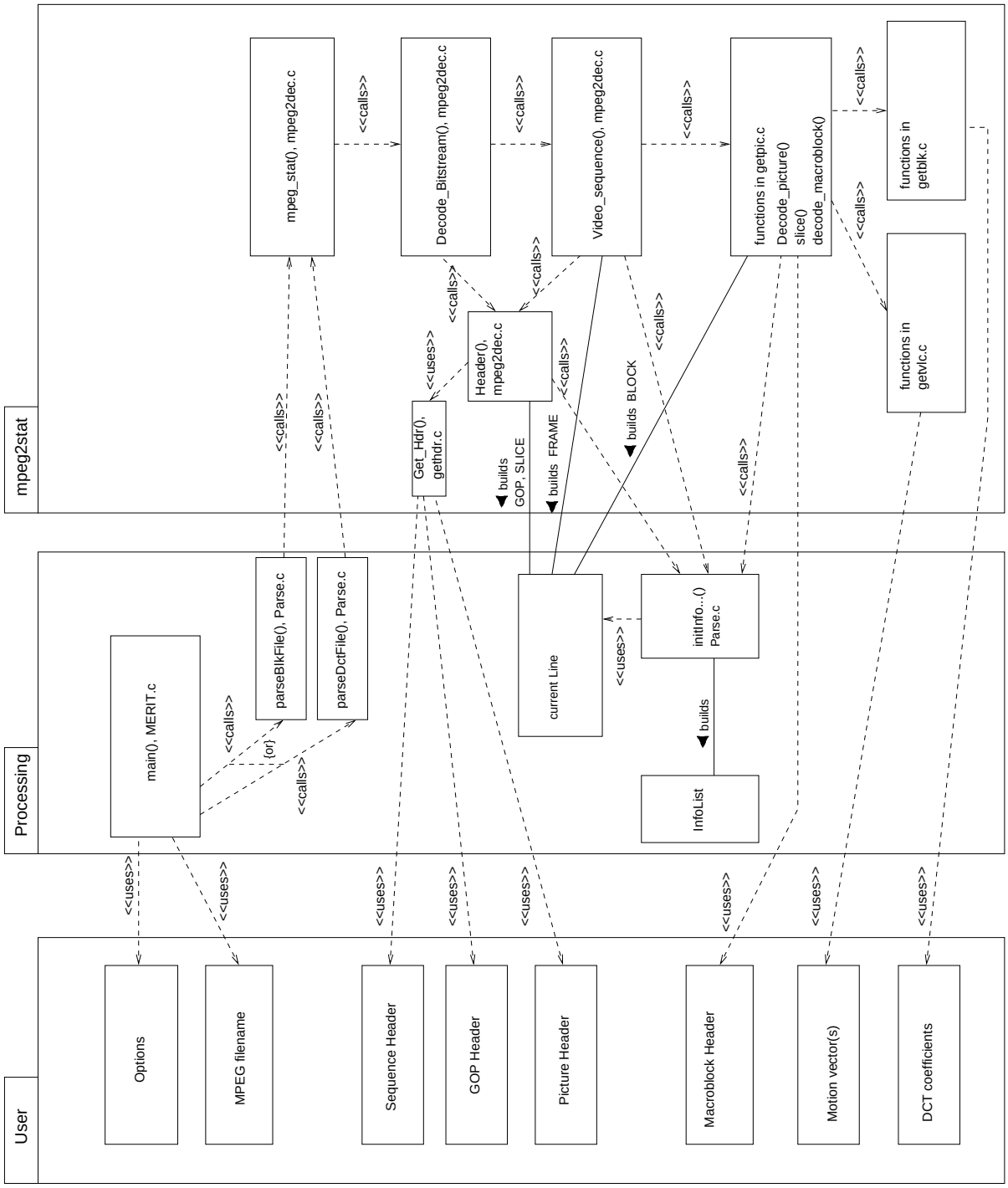


Figure 12: InfoList building collaboration diagram.

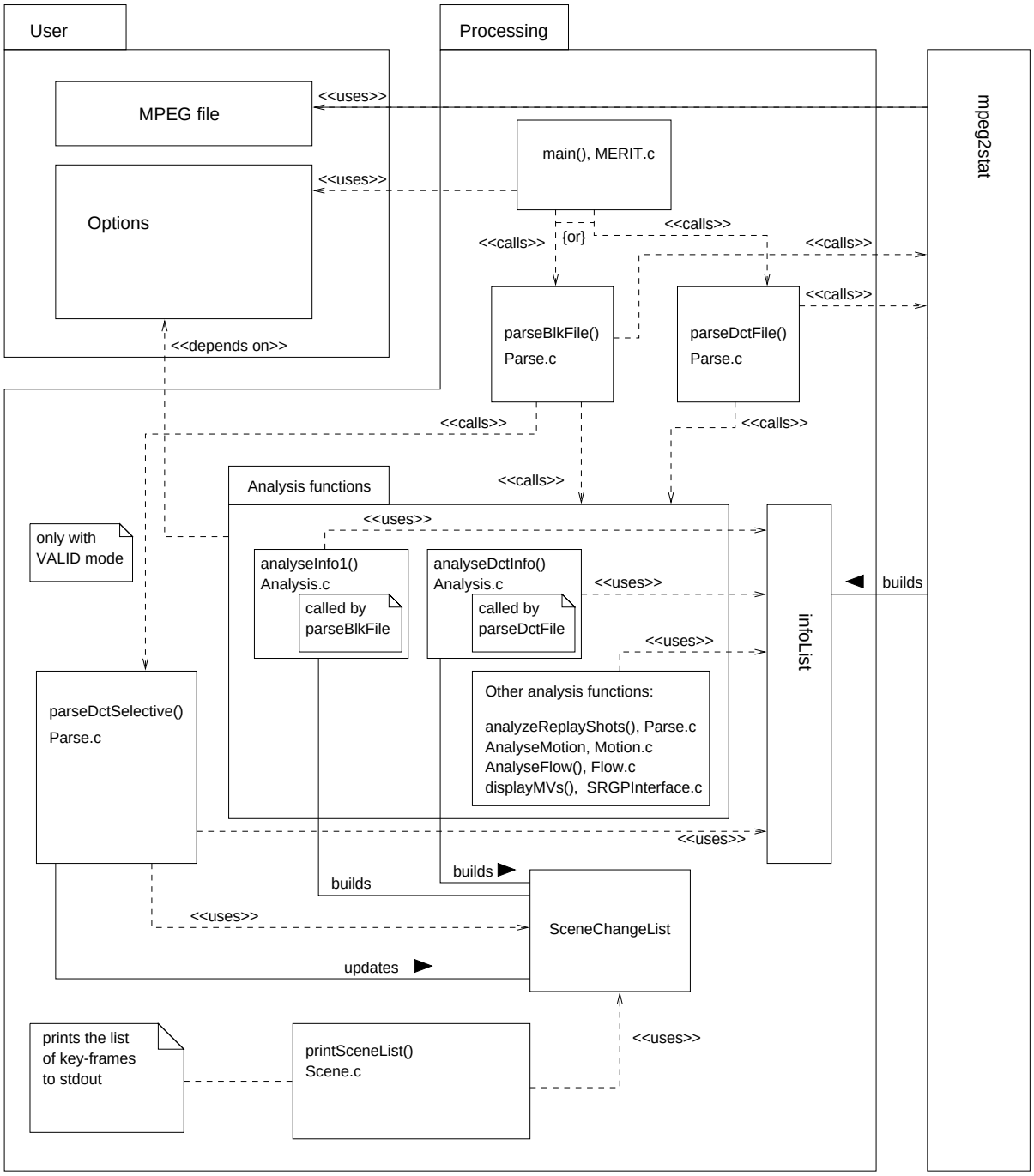


Figure 13: Processing collaboration diagram.

## References


[1] B. G. Haskell, A. Puri, and A. N. Netravali,"Digital Video: An Introduction to MPEG-2", Chapman and Hall, 1997.

[2] V. Kobla, MERIT Documentation, MERIT Software Version 3.1.

[3] V. Kobla and D. S. Doermann, "Indexing and retrieval of MPEG compressed video", Journal of Electronic Imaging, Vol. 7, pp. 294–307, 1998.

[4] V. Kobla, D. S. Doermann, and K. I. Lin, "Archiving, indexing and retrieval of video in the compressed domain", Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems, Vol. 2916, pp. 78–89, 1996.

[5] V. Kobla, D. S. Doermann, and A. Rosenfeld, "Compressed Domain Video Segmentation", CfAR Technical Report CAR-TR-839 (CS-TR-3688), 1996.

[6] V. Lo, "A Beginner's Guide for MPEG-2 Standards", `http://www.fh-friedberg.de/fachbereiche/e2/telekom-labor/zinke/mk/mpeg2beg/beginnzi.htm`

[7] T. Sikora, "MPEG Digital Video Coding Standards", Digital Electronics Consumer Handbook, McGraw Hill Company, Ed. R. Jurgens, `http://wwwam.HHI.DE/mpeg-video/papers/sikora/mpeg1_2/mpeg1_2.htm`

[8] P. N. Tudor, "MPEG-2 video compression", Electronics and Communication Engineering Journal, 1995, `http://www.bbc.co.uk/rd/pubs/papers/paper_14/paper_14.html`

[9] "Generic Coding of Moving Pictures and Associated Audio Information: Video", ISO/IEC 13818-2: Draft International Standard, 1994.

[10] V. Kobla, "MERIT Video Segmentation", `http://documents.cfar.umd.edu/LAMP/Media/Presentations/MERITtheory/index.htm`

[11] V. Kobla, "MERIT MPEG Encoded Retrieval and Indexing Toolkit", `http://documents.cfar.umd.edu/LAMP/Media/Presentations/MERITver3talk/index.htm`

[12] The MPEG Home Page — Moving Picture Experts Group Website, `http://drogo.cselt.it/mpeg`